C++程序设计第 6 次作业 大数据 2101 班 李嘉鹏 U202115652 2022 年 11 月 9 日

C++模拟试卷-7

一、选择题(每小题 3 分，共 15 分)

1. 对于如下程序：

```
#include <stdio.h>
struct A {
    virtual int f( ) { printf("A"); }
    virtual int g( ) { printf("B"); }
}a,*p;
struct B: A {
    int f( ) { A::f( ); printf("C"); }
    int g( ) { printf("D"); }
}b;
void main( ) { p=&b; p->f( ); p->g( ); }
```

则程序的输出为__C__：

A. AB　　B. CD

C. ACD　　D. ABCD

2. 如下程序：

```
#include <stdlib.h>
struct A { A( ) { } } a;
void main(  ) { A b; exit(0 ); }
```

则关于对象 a, b 的析构，如下哪个叙述正确__B__：

A. 无析构函数都没有析构;　　B. a 析构了但是 b 没析构;

C. b 析构了但是 a 没析构;　　D. 都析构了;

3. 对于定义“const char *&g( );”，如下哪个语句是错误的__A__：

A. g( ) = "abcde";　　B. *g( ) = 'A';

C. const char *p = g( );　　D. const char *&q = g( );

4. 对于如下定义：

```
struct A { virtual int f( ) { return 1; } } a;
struct B: A {
    int f( ) const { return 2; }
    int f( ) volatile { return 3; }
    int f( ) const volatile { return 4; }
```

```
} c;
int main(int argc, char *argv[ ]) { A *p=&c; return p->f( ); }
```

主函数 main 的返回值是__A__:

A. 1　　B. 2

C. 3　　D. 4

5. 关于运算符重载的叙述哪个正确__D__:

A. 可以改变优先级和结合性　　B. 可以改变优先级，不能改变结合性

C. 不能改变优先级，可改变结合性　　D. 都不能改变

二、在作用域运算最多允许一层（即不允许两层运算如 A::B::c 及两层以上运算）的情况下，指出各类可访问的成员及其访问权限 (20 分)。

```
class A {                  |A 的成员
    int   a;               |
protected:                 |
    int   b, e;            |
public:                    |
    int   c, d;            |
};                         |
class B: protected A {     |B 的成员
    int   d;               |
protected:                 |
    int   c, e;            |
public:                    |
     A::c;                 |
    int   f, b;            |
};
class C: public A {        |C 的成员
    int   g;               |
protected:                 |
    int   h, d;            |
public:                    |
    int   i;               |
};                         |
```

```
struct D: protected B, C {  | D 的成员
    int   j;                |
protected:                  |
    int   k;                |
public:                     |
    int   n, d;             |
};
```

答：A 类：

private：a

protected：b，e

public：c，d

B 类：

private：d

protected：c，e，A::b，A::e，A::c，A::d

public：f，b

C 类：

private：g

protected：h，d，A::b，A::e

public：i，A::c，A::d

D 类：

private：

protected：k，B::c，B::e，C::h，C::d

public：j，n，d，C::i

三、指出 main 中每行的输出结果 (前四题每题 3 分，后两题每题 4 分，共 20 分)

```
#include <iostream.h>
struct A { A( ) { cout<<'A'; } };
struct B { B( ) { cout<<'B'; } };
struct C: virtual A { C( ) { cout<<'C'; }} ;
struct D: B, virtual C { D( ) { cout<<'D'; } };
struct E: virtual A, virtual D {
    D   d;
    E( ): A( ) { cout<<'E'; }
};
struct F: virtual C, B, virtual D, virtual E {
```

```
    D d;   E e;
    F( ) { cout<<'F'; }
};
void main( ) {
    A   a;   cout<<'\n';   //输出=A
    B   b;   cout<<'\n';   //输出=B
    C   c;   cout<<'\n';   //输出=AC
    D   d;   cout<<'\n';   //输出=ACBD
    E   e;   cout<<'\n';   //输出=ACBDACBDE
    F   f;   cout<<'\n';   //输出=ACBDACBDEBACBDACBDACBDEF
}
```

四、指出以下程序的语法错误及其原因 (每错约 1 分，共 15 分)

```
class A {
    int &a;
protected:
    const int &b;
    ~A( ) { }
public:
    int   c;
    virtual A&   (*g)( );  错，成员函数指针不能定义为 virtual
    A(int x) { b=x; };     错，b 是 const int 变量，要放在初始化列表中初始化
} a=(1, 2, 3);             错，A 类的析构函数~A()是 protected，生成对象后不可访问析构函数
class B: A {
    int   d;
public:
    A::b;
    static int operator( )(int) { return 3; };     错，operator 函数不能为静态成员函数
    B(int x, int y, int z):A(x) { d=x+y+z; };
}b(2, 3, 7);
struct C: B {
    int z;
protected:
```

```
    virtual ~C( ) {   };
}c;
void   main( ) {
    int   *A::*p=&c.z;      错，c.z 不是 A 类对象，其地址不能赋给 int *A::*类对象
    int   i=a.b;            错，a.b 为 private，不可访问
    i=a;                    错，A 类内没有定义 int 运算符重载函数
    i=b.b;
    i=c.d;                  错，B::b 为 private，无法被 C 类对象继承，不可访问
    i=b.*p;                 错，类的成员指针不可参与运算
    return 1;               错，void main 函数没有返回值
}
```

五、请填入学号最后一位十进制数字，指出 main 函数中变量 i 在每条赋值语句执行后的值 (每小题 2.5 分，共 15 分)

```
int    x = 2____,   y=x+3;
struct A {
    int   x;
    static int &y;
public:
    operator int( ) const { return x+y; }
    int &v(int &x) {
        for(int y=1; x<201; x^=y, y++)
            if(x>200) { x-=31; y-=2;}
         return ++x;
    }
    A &operator++( ){ ++x; ++y; return *this; }
    A(int x=::x+2, int y=::y+3){ A::x=x;   A::y=y; }
};
int & A::y=::x;
void main( ){
    A   a(2, 3),   b(a),   c;
    int   i,   &j=i,   A::*p=&A::x;
    i=a.y;                  //i=8
    j=a.x++;                //i=2
    i=a.*p;                 //i=3
```

```
    i=++a;          //i=13
    i=b.y+::y;      //i=14
    (b.v(i)=2)+=3;  //i=5
}
```

六、N 个顶点的无向图 MAP 最多有 N*(N-1)条边，设顶点的编号为 0, 1, …, N-1，每条边由其中任意两个顶点连接而成，试定义如下无向图类中的成员函数。(每小题 2.5 分，共 15 分)

```
class MAP {
    int (*const e)[2];    //边集指针 e,边 x 的顶点为 e[x][0]和 e[x][1]
    const int n;          //图的顶点个数
    int c;                //图实际已有的边的个数
public:
    MAP(int n);           //图最多 n 个顶点，假设图初始时无边
    MAP(const MAP& m);    //深拷贝构造函数
    MAP& operator=(const MAP& m);   //深拷贝赋值函数
    MAP& operator( )(int v0, int v1);   //连接顶点 v0 和 v1 成边,设 v0<v1
    int(*operator[ ](int x))[2];        //取图中的边 x
    ~MAP( );                            //析构函数
};
```

成员函数定义如下：

```
MAP::MAP(int n) :n(n) {
    c = 0;
    *(int**)&e = new ((int*)[2])[n * (n - 1) / 2];
}

MAP::MAP(const MAP& m) :n(m.n) {
    c = m.c;
    *(int**)&e = new ((int*)[2])[n * (n - 1) / 2];
    for (int i = 0; i < c; i++)
    {
        e[i][0] = m.e[i][0];
        e[i][1] = m.e[i][1];
    }
}

MAP& MAP::operator=(const MAP& m) {
    *(int*)&n = m.n;
    c = m.c;
    for (int i = 0; i < c; i++)
    {
```

```
            e[i][0] = m.e[i][0];
            e[i][1] = m.e[i][1];
        }
        return *this;
}

MAP& MAP::operator( )(int v0, int v1) {
        for (int i = 0; i < c; i++)
            if (e[i][0] == v0 || e[i][0] == v1)
                return *this;
        c++;
        e[c - 1][0] = v0;
        e[c - 1][1] = v1;
        return *this;
}

int(*MAP::operator[ ](int x))[2] {
        if (x >= c)
        {
            printf("输入不合法！");
            return NULL;
        }
        return e + x;
}

MAP::~MAP() {
        if (e) {
            delete[]e;
            *(int*)&n = 0;
            c = 0;
        }
}
```